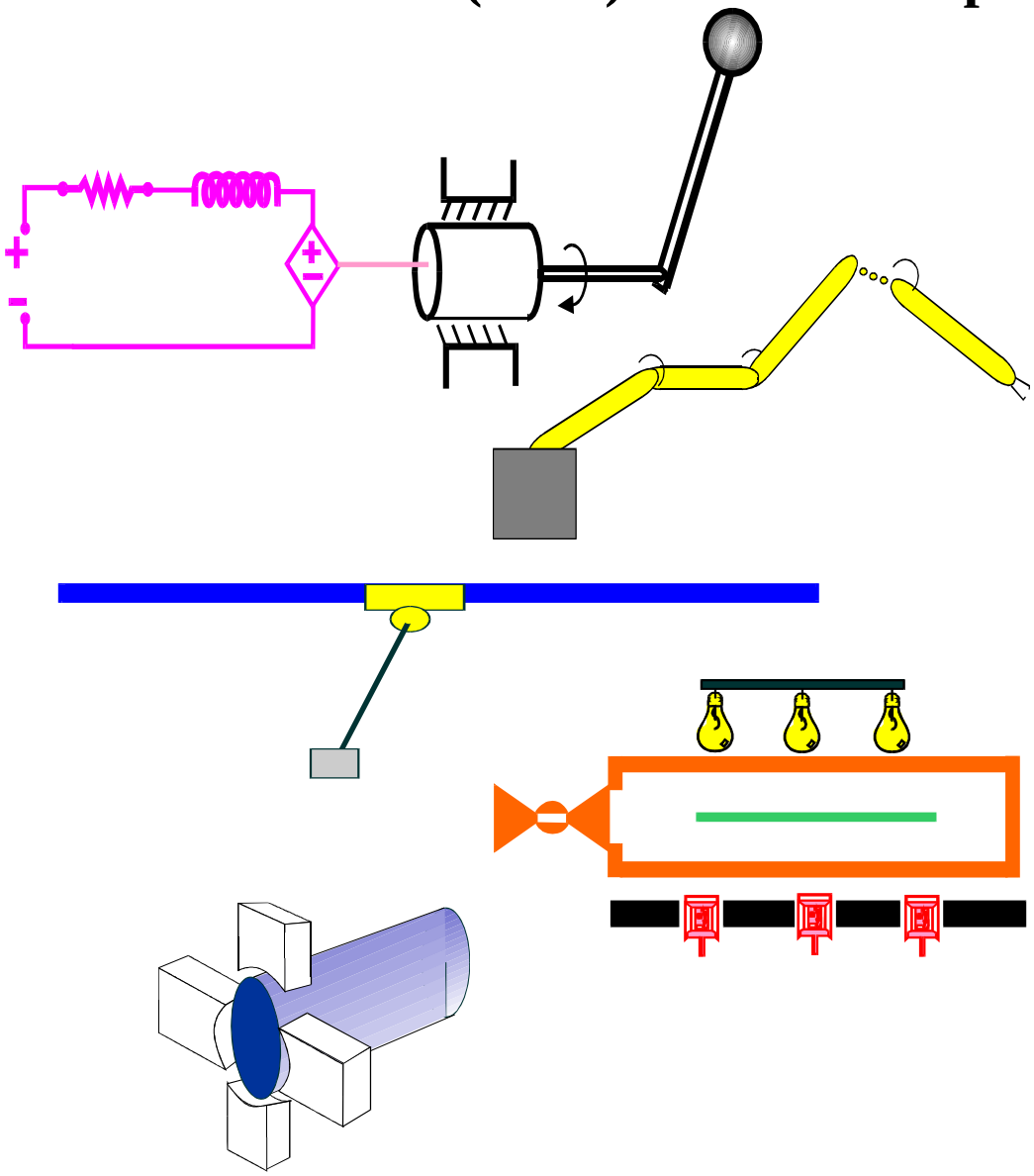
# Clemson University
# College of Engineering and Science
# Control and Robotics (CRB) Technical Report



Number: CU/CRB/3/2/06/#1

Title: Neural Network Grasping Controller for Continuum Robots

Authors: D. Braganza, D. M. Dawson, I. D. Walker and N. Nath

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **2006** | | **00-00-2006 to 00-00-2006** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Neural Network Grasping Controller for Continuum Robots** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>**Clemson University,Department of Electrical & Computer Engineering,Clemson,SC,29634-0915** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES
**The original document contains color images.**

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | |
| **unclassified** | **unclassified** | **unclassified** | | **11** | |

# Neural Network Grasping Controller for Continuum Robots

D. Braganza, D. M. Dawson, I. D. Walker and N. Nath

*Abstract*— **Continuum or hyper-redundant robots are robots which exhibit behavior similar to biological trunks, tentacles and snakes. Unlike traditional robots, continuum robot manipulators do not have rigid joints, hence the manipulators are compliant, extremely dexterous, and capable of dynamic, adaptive manipulation in unstructured environments; however, the development of high-performance control algorithms for these manipulators is a challenging problem. In this paper, we present an approach to whole arm grasping control for continuum robots. The grasping controller is developed in two stages; high level path planning for the grasping objective, and a low level joint controller using a neural network feedforward component to compensate for dynamic uncertainties. These techniques are used to enable whole arm grasping without using contact force measurements and without using a dynamic model of the continuum robot. Experimental results using the OCTARM, a soft continuum robotic manipulator are included to illustrate the efficacy of the proposed low level joint controller.**

## I. Introduction

Continuum or hyper-redundant robot manipulators are manipulators which exhibit behavior similar to biological trunks, tentacles and snakes [1], [2]. Unlike traditional rigid link robots, continuum robot manipulators do not have rigid joints, also the increased number of degrees of freedom give the manipulator some very useful properties. The manipulators are flexible, compliant, extremely dexterous and capable of dynamic adaptive manipulation in unstructured environments. Due to these inherent properties of soft continuum robot manipulators, they are uniquely suited to perform whole arm grasping. Whole arm manipulation [3] is the term used to describe the ability of the manipulator to grasp an object using its entire body, as compared to fingertip grasping performed by traditional robotic grippers and hands. Whole arm grasping is performed by allowing the robot manipulator to make contact with the object in a snake or tentacle-like manner, using portions of the manipulator itself to wrap around the object and grasp it. Several researchers including [3], [4] and others have pointed out the advantages of whole arm grasping, some of which are, increased load capacity and the ability to grasp objects of various dimensions. These capabilities can be used in many applications, including search and rescue, underwater and space exploration. The grasping techniques presented in most of the previous work requires either that the geometry of the object and the constraint forces to be known *a priori* [5], or that the contact forces be measurable using some type of force sensor [6], [7].

The development of high performance model based control algorithms for continuum manipulators is a challenging problem for several reasons. For example, since the arms must be modeled as continuous curves, the kinematic and dynamic models are difficult to derive. Also the manipulator body is soft and flexible which makes accurate control difficult. Several researchers have proposed kinematic control techniques for continuum manipulators, for example see [8], and the references therein. A set-point controller for a variable length manipulator based on an artificial potential function method which does not require the dynamic model was presented in [9]. Various other techniques have been suggested for tracking control of continuum manipulators. In [10], sliding mode and impedance control techniques for hyper-flexible manipulators were presented. In [11], the authors present a trajectory tracking control of snake robots based on its dynamic model. In [12], a fuzzy control method was presented. Shape tracking control, where the manipulator follows a desired shape prescribed by a time-varying spatial curve was considered by [13]. All of the aforementioned techniques but [9] require the dynamic model of the manipulator be known.

In this paper, a path planner is presented for whole arm grasping which does not require the constraint forces to be known *a priori* and also eliminates the requirement for contact force sensing. The path planner is then fused with a low level joint controller that uses a neural network feedforward component to compensate for the unknown dynamics of the continuum robot manipulator. Both the planner and controller are designed such that force measurements are not required. The design of the neural network is based on the augmented back propagation algorithm [14]. Specifically, the neural network is used to compensate for the nonlinear uncertain dynamics of the continuum robot manipulator while a nonlinear feedback controller [15] is used to provide semi-global asymptotic tracking. The advantage of the proposed control scheme compared to previous works is that semi-global asymptotic tracking can be proved, whereas most previous results for neural network control of robot manipulators [16]–[18] only prove ultimate boundedness of the tracking error.

The remainder of the paper is organized as follows, in section II, the kinematic and dynamic models for the continuum robot are presented. In section III, the high level path planning for whole arm grasping is presented. In section IV, the low level joint control objective is defined, and the design of the low level controller with the neural network

This work is supported in part by a DOC Grant, an ARO Automotive Center Grant, a DOE Contract, a Honda Corporation Grant and a DARPA Contract.

The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634-0915. dbragan@clemson.edu

feedforward component is presented. To demonstrate the performance of the proposed controller with the neural network feedforward component, the controller was tested on the OCTARM, a soft continuum robotic manipulator for a sinusoidal trajectory. Experimental results both with and without the neural network feedforward component are presented in section V to illustrate the effectiveness of the proposed control strategy.

## II. System Models

### A. Kinematic Model

The forward kinematic model for an $n$-segment continuum robot can be developed as follows

$$x_n = f_n(q) \tag{1}$$

where $x_n(t) \in \mathbb{R}^p$ represents the robot end-effector's task-space vector, $q(t) \in \mathbb{R}^n$ denotes the joint position, and $f_n(q) \in \mathbb{R}^p$ denotes the forward kinematics of the robot. Note for continuum robots, $q(t)$ is a vector of curvatures and extensions of the robot sections, for more information see [19]. The velocity kinematics for the robot can be developed as follows

$$\dot{x}_n = J_n(q)\dot{q}(t) \tag{2}$$

where $\dot{x}_n(t) \in \mathbb{R}^p$ represents the task-space velocity, $\dot{q}(t) \in \mathbb{R}^n$ denotes the joint velocity, and $J_n(q) \triangleq \dfrac{\partial f_n(q)}{\partial q} \in \mathbb{R}^{p\times n}$ denotes the robot Jacobian. Recently in [19], the authors have developed a general method for determining the kinematics of continuum robots. This approach enables the Cartesian position and orientation of the end effector and the robot Jacobian to be computed in real-time.

*Assumption 1: The kinematic model for the continuum robot is known, and all kinematic singularities are avoided such that $J(q)$ is always non-singular.*

### B. Dynamic Model

From a review of the literature, it is evident that there have been few published results pertaining to the dynamic modeling of continuum robot arms. Some of the dynamic models which have been developed include [20], [21], where the planar model of the manipulator was considered, and [22], where the authors develop a 3D dynamic model for a constant length, non-extensible continuum manipulator. As such, the complete dynamic modeling of variable length continuum robot arms remains an open research area. In [22], the developed dynamic model was shown to satisfy the familiar property that the continuum manipulators inertia matrix is symmetric and positive definite. In this development, we will assume that the dynamic model of a 9-joint continuum robot manipulator can be described by the following expression

$$M(q)\ddot{q} + N(q,\dot{q}) = u \tag{3}$$

where $M(q) \in \mathbb{R}^{9\times 9}$ represents the inertia matrix, $N(q,\dot{q}) \in \mathbb{R}^9$ represents the remaining dynamic terms, $u(t) \in \mathbb{R}^9$ represents the input torque vector, and $q(t)$, $\dot{q}(t)$, $\ddot{q}(t) \in \mathbb{R}^9$ represent the joint position, velocity and acceleration respectively.

The subsequent development is based on the following assumptions

*Assumption 2: The manipulators joint position $q(t)$ and joint velocity $\dot{q}(t)$ are measurable.*

*Assumption 3: The dynamic terms denoted by $M(q)$ and $N(q,\dot{q})$ are unknown nonlinear functions of $q(t)$ and $\dot{q}(t)$ which are second order differentiable and satisfy the following properties*

$$M(\cdot), \dot{M}(\cdot), \ddot{M}(\cdot) \in \mathcal{L}_\infty \quad \text{if } q(t), \dot{q}(t), \ddot{q}(t) \in \mathcal{L}_\infty \tag{4}$$

$$N(\cdot), \dot{N}(\cdot), \ddot{N}(\cdot) \in \mathcal{L}_\infty \quad \text{if } q(t), \dot{q}(t), \ddot{q}(t), \dddot{q}(t) \in \mathcal{L}_\infty. \tag{5}$$

*Assumption 4: The inertia matrix $M(q)$ is symmetric and positive-definite, and satisfies the following inequalities*

$$m_1 \|\xi\|^2 \leq \xi^T M(q)\xi \leq m_2 \|\xi\|^2 \qquad \forall \xi \in \mathbb{R}^9 \tag{6}$$

*where $m_1$, $m_2 \in \mathbb{R}$ are positive constants, and $\|\cdot\|$ denotes the standard Euclidean norm.*

## III. High Level Path Planning

Whole arm grasping can be achieved by integrating the path planner and the controller such that two tasks, robot *end-effector positioning* and robot *body self-motion positioning*, are accomplished simultaneously [23]. The *end-effector positioning* controller forces the end-effector to follow a path around the object which in turn, forces the robot's body to wrap itself around the object to be grasped. The *body self-motion positioning* controller "repels" the body of the manipulator away from the object while the end-effector moves around the object.

### A. Kinematic Planning

To facilitate the kinematic planning, the pseudo-inverse of the manipulator Jacobian denoted by $J_n^+(q) \in \mathbb{R}^{n\times p}$ is defined as follows

$$J_n^+ \triangleq J_n^T \left(J_n J_n^T\right)^{-1} \tag{7}$$

where $J_n^+(q)$ satisfies the following equality

$$J_n J_n^+ = I_p \tag{8}$$

where $I_p \in \mathbb{R}^{p\times p}$ is the standard identity matrix. As shown in [24], the pseudo-inverse defined by (7) satisfies the Moore-Penrose conditions given below

$$\begin{array}{ll} J_n J_n^+ J_n = J_n & J_n^+ J_n J_n^+ = J_n^+ \\ \left(J_n^+ J_n\right)^T = J_n^+ J_n & \left(J_n J_n^+\right)^T = J_n J_n^+. \end{array} \tag{9}$$

In addition to the above properties, the matrix $(I_n - J_n^+ J_n)$ satisfies the following useful properties

$$\begin{array}{c} \left(I_n - J_n^+ J_n\right)\left(I_n - J_n^+ J_n\right) = I_n - J_n^+ J_n \\ \left(I_n - J_n^+ J_n\right)^T = \left(I_n - J_n^+ J_n\right) \\ J_n\left(I_n - J_n^+ J_n\right) = 0 \\ \left(I_n - J_n^+ J_n\right) J_n^+ = 0 \end{array} \tag{10}$$

where $I_n \in \mathbb{R}^{n\times n}$ is the standard identity matrix.

Based on (2) and the above properties the kinematic planner, denoted by $U(t) \in \mathbb{R}^n$, that enables the whole arm grasping objective is designed as follows

$$U(t) \triangleq J_n^+ U_e + \left(I_n - J_n^+ J_n\right) U_m \tag{11}$$

where $U_e(t) \in \mathbb{R}^p$ is the *end-effector positioning* controller, and $U_m(t) \in \mathbb{R}^n$ is the robot *body self-motion* controller. The objective of the *end-effector positioning* controller is to force the end-effector to track a desired trajectory encompassing the surface of the object to be grasped. We will utilize a task space velocity field, [25], for the end-effector positioning because it more effectively penalizes the end-effector for leaving the contour and does not exhibit the radial reduction phenomenon [25], [26]. For example, when the object to be grasped is circular, the velocity field can be designed to generate a desired trajectory which forces the end-effector to spiral inwards, toward, and around the surface of the object.

The *end-effector positioning* controller is designed [23] as

$$U_e \triangleq \vartheta(x_n) + K_e e + k_n \left\| \frac{\partial V(x_d)}{\partial x_d} \right\|^2 \rho^2(x_n, x_d)\, e \quad (12)$$

where $\vartheta(x_n) \in \mathbb{R}^p$ is a task-space velocity field which encircles the object to be grasped, $K_e \in \mathbb{R}^{p \times p}$ is a positive definite diagonal gain matrix, $k_n \in \mathbb{R}^+$ is a scalar gain parameter, $e(t) \in \mathbb{R}^p$ is the error between the desired and actual task space position and is defined as follows

$$e \triangleq x_d - x_n, \quad (13)$$

where $x_d(t) \in \mathbb{R}^p$ is the desired task-space position, and $x_n(t)$ was introduced in (1). In (12), $V(x_d) \in \mathbb{R}$ is a first order differentiable, nonnegative function, and $\rho(\cdot) \in \mathbb{R}$ is a known positive function that is assumed to be bounded provided $x_n(t)$ and $x_d(t)$ are bounded. The desired task space velocity trajectory is defined as

$$\dot{x}_d(t) \triangleq \vartheta(x_n) \quad (14)$$

where $\vartheta(x_n)$ is the velocity field generated by the task-space position $x_n(t)$. Refer to [23] for more details of this development.

The objective of the *body self-motion positioning* controller is to "repel" the end-effector and body of the manipulator away from the object to be grasped, while the end-effector moves around the object. This repulsion-like property facilitates obstacle avoidance and removes the "slack" from the body of the manipulator as the robot moves into the grasping position. Following this line of reasoning, the *body self-motion positioning* controller $U_m(t) \in \mathbb{R}^n$ of (11), is designed [23] as follows

$$U_m \triangleq -k_m \left[ J_s \left( I_n - J_n^+ J_n \right) \right]^T y_a \quad (15)$$

where $k_m \in \mathbb{R}^+$ is a control gain, $J_s \in \mathbb{R}^{1 \times n}$ is a Jacobian-like vector, $I_n \in \mathbb{R}^{n \times n}$ is the standard identity matrix, and $y_a(t) \in \mathbb{R}$ is an auxiliary scalar signal encoding geometric information about the object's surface and it's relationship to the manipulator joint positions. This type of geometric encoding keeps the body of the manipulator away from the object during the initial phases of grasping.

For whole arm grasping, a specific auxiliary signal $y_a(t)$ is designed as follows

$$y_a \triangleq \sum_{i=1}^{m} h_{ai}(x_i) \quad (16)$$

where $m$ is the number of sections of the redundant manipulator, $x_i = \begin{bmatrix} \bar{x}_{i1} & \bar{x}_{i2} & \dots & \bar{x}_{ip} \end{bmatrix}^T \in \mathbb{R}^p$ is the Euclidean-space coordinate for the $i^{th}$ section, and $h_{ai}(x_i) \in \mathbb{R}$ is the repulsion function for the $i^{th}$ section that encodes geometric information about the surface of the object with respect to the $i^{th}$ section's Euclidean position. The repulsion function $h_{ai}(x_i)$ is defined as follows

$$h_{ai}(x_i) = k_{hi} \exp\left(-\alpha_i \beta_i^2(x_i)\right), \quad \forall\, i = 1, .., m \quad (17)$$

where $k_{hi},\ \alpha_i \in \mathbb{R}^+$ are constants, and $\beta_i(x_i) \in \mathbb{R}$ is the section specific geometric function. The function $\beta_i(x_i)$ should be designed to be positive when the manipulator is not touching the object. For example, given a spherical object in three dimensional Euclidean-space, $\beta_i(x_i)$ could be defined as follows

$$\beta_i(x_i) \triangleq (\bar{x}_{i1} - \bar{x}_{c1})^2 + (\bar{x}_{i2} - \bar{x}_{c2})^2 + (\bar{x}_{i3} - \bar{x}_{c3})^2 - r_o^2$$

where $\bar{x}_{c1}, \bar{x}_{c2}, \bar{x}_{c3}, r_o \in \mathbb{R}$ are the Euclidean coordinates of the center of the spherical object and its radius, respectively. For more details of this development refer to [23].

### *B. Fusing the Planner with the Controller*

To fuse the high level path planner with the low level joint controller, we use a desired trajectory generator which ensures that the desired trajectories for the manipulator's joints are bounded. The structure of the desired trajectory generator is motivated by the choice of the low level controller (25), which requires the desired trajectory be bounded up to its fourth derivative.

To ensure that the desired joint space velocity trajectory is bounded, we could use the following expression

$$\dot{q}_d(t) \triangleq sat(U(t)) \quad (18)$$

where $U(t)$ was defined in (11), $\dot{q}_d(t) \in \mathbb{R}^9$ are the desired joint velocities, $sat(\xi) \in \mathbb{R}^n$ is defined as $sat(\xi) = [sat(\xi_1), \cdots, sat(\xi_n)]^T\ \forall \xi = [\xi_1, \cdots, \xi_n]^T \in \mathbb{R}^n$ where $sat(\xi_i) \in \mathbb{R}\ \forall i = 1, \cdots, n$ is the following saturation function

$$sat(\xi_i) = \begin{cases} -\xi_{min} & \text{if} \quad \xi_i \le -\xi_{min} \\ \xi_i & \text{if} \quad \xi_i > -\xi_{min} \quad \text{or} \quad \xi_i < \xi_{max} \\ \xi_{max} & \text{if} \quad \xi_i \ge \xi_{max} \end{cases}$$

where $\xi_{min}, \xi_{max} \in \mathbb{R}^+$ are constants. If (18) is used to generate the desired trajectory, we cannot prove that the desired joint trajectory, $q_d(t) \in \mathbb{R}^9$ is bounded, so we could use the following filtering operation

$$q_d(s) \triangleq \frac{1}{\left(\frac{s}{\epsilon} + 1\right)} sat(U(t)) \quad (19)$$

where $s \in \mathbb{C}$ is the standard Laplace variable, and $\epsilon \in \mathbb{R}^+$ is a small constant. However, in the case of (19), we cannot prove that the higher order derivatives of $q_d(t)$ will remain bounded. So the desired trajectory generated for $q_d(t)$ is modified further in the final form given by

$$q_d(s) \triangleq \frac{1}{\left(\frac{s}{\epsilon} + 1\right)\left(\frac{s}{\kappa} + 1\right)^3} sat(U(t)) \quad (20)$$

where $\kappa \in \mathbb{R}^+$ is large constant. From (20), it is clear that $q_d(t)$, $\dot{q}_d(t)$, $\ddot{q}_d(t)$, $\dddot{q}_d(t)$, and $\ddddot{q}_d(t) \in \mathcal{L}_\infty$.

## IV. Low level joint control objective

The low level control objective is to design a continuous controller which provides asymptotic tracking of the manipulator joint position and the desired trajectory in the sense that

$$q(t) \rightarrow q_d(t) \text{ as } t \rightarrow \infty. \tag{21}$$

To quantify the control objective, an error signal, denoted by $e_1(t) \in \mathbb{R}^9$, is defined as follows

$$e_1 \triangleq q_d - q. \tag{22}$$

Furthermore, an auxiliary tracking error signal $e_2(t) \in \mathbb{R}^9$ is defined as follows

$$e_2 \triangleq \dot{e}_1 + \lambda_1 e_1 \tag{23}$$

where $\lambda_1 \in \mathbb{R}^+$ is a control gain. For the closed loop error system development, we define a filtered tracking error signal $r(t) \in \mathbb{R}^9$ as follows

$$r \triangleq \dot{e}_2 + \lambda_2 e_2 \tag{24}$$

where $\lambda_2 \in \mathbb{R}^+$ is a control gain.

The dynamic model of the continuum robot is a nonlinear uncertain system; hence, the strategy developed by Xian et. al. [15] can be used for the joint level controller. This controller is chosen because it is continuous, it does not require the dynamic model of the manipulator or contact forces to be known and yet it provides semi-global asymptotic tracking. Specifically, the control objective described in (21) can be met with the following controller [15]

$$\begin{aligned} u(t) \triangleq\ & (K_s + I)e_2(t) - (K_s + I)e_2(t_0) + \int_{t_0}^{t} \hat{f}(\tau)d\tau \\ & + \int_{t_0}^{t} \left(\lambda_2 (K_s + I)e_2(\tau) + \beta sgn(e_2(\tau))\right) d\tau \end{aligned} \tag{25}$$

where $u(t) \in \mathbb{R}^9$ is the control input defined in (3), $\lambda_2 \in \mathbb{R}^+$ is a control gain $K_s$, $\beta \in \mathbb{R}^{9\times 9}$ are positive definite diagonal control gain matrices, $\hat{f}(t) \in \mathbb{R}^9$ is the neural network feedforward component and $sgn(\cdot) : \mathbb{R}^9 \mapsto \mathbb{R}^9$ denotes the vector signum function defined as $sgn(\xi) = [sgn(\xi_1), \cdots, sgn(\xi_9)]^T \ \ \forall \xi = [\xi_1, \cdots, \xi_9]^T \in \mathbb{R}^9$. The controller presented in (25), provides semi-global asymptotic convergence of the joint position tracking error, (i.e. $\|e_1(t)\| \rightarrow 0$ as $t \rightarrow \infty$). For a detailed analysis of the controller the reader is referred to [15].

*Remark 1: The design of the neural network feedforward component, $\hat{f}(t)$, is presented in the subsequent section. The only restriction imposed on the neural network component by the selection of the controller (25) is that $\hat{f}(t) \in \mathcal{L}_\infty$.*

### A. Neural Network Feedforward Design

The neural network feedforward component $\hat{f}(t) \in R^9$ is computed using a two layer network with 15 neurons[1]. The weights are computed using a modified version of the back propagation algorithm presented in [14]. Given Remark 1, an important consideration regarding the design of the neural network feedforward component is that the output from the neural network must always be bounded (i.e. $\hat{f}(t) \in \mathcal{L}_\infty$). To this end the neural network component is defined as follows

$$\hat{f} = \hat{W}^T \bar{\sigma}\left(\hat{V}^T x\right). \tag{26}$$

where $\hat{W}(t) \in \mathbb{R}^{15\times 9}$ and $\hat{V}(t) \in \mathbb{R}^{37\times 15}$ are estimated weight matrices, and $x(t) \in \mathbb{R}^{37}$ is the input vector to the neural network which is selected as

$$x = \begin{bmatrix} 1, & q_d^T, & \dot{q}_d^T, & \ddot{q}_d^T, & \dddot{q}_d^T \end{bmatrix}^T \tag{27}$$

where $q_d(t), \dot{q}_d(t), \ddot{q}_d(t), \dddot{q}_d(t)$ were previously defined. The vector activation function $\bar{\sigma}(\cdot) \in \mathbb{R}^{15} \mapsto \mathbb{R}^{15}$ is defined as follows

$$\bar{\sigma}(\omega) = [\sigma(\omega_1), \sigma(\omega_2), \cdots, \sigma(\omega_{15})]^T \tag{28}$$

where $\omega = [\omega_1, \omega_2, \cdots, \omega_{15}]^T$ and $\sigma(s) : \mathbb{R} \mapsto \mathbb{R}$ is the sigmoid activation function defined as

$$\sigma(s) = \frac{1}{1 + \exp(-s)}. \tag{29}$$

The gradient of the vector activation function, denoted by $\bar{\sigma}'(\cdot) \in \mathbb{R}^{15\times 15}$ can be expressed in closed form as follows, [14]

$$\bar{\sigma}(\omega)' = diag\{\bar{\sigma}(\omega)\}\left[I - diag\{\bar{\sigma}(\omega)\}\right]. \tag{30}$$

If we were to design the weight update laws according to the augmented backpropagation algorithm [14], we would use the following update rule

$$\dot{\hat{W}} = -\kappa F \|r\| \hat{W} - F\bar{\sigma}'(\cdot)\hat{V}^T x r^T + F\bar{\sigma}(\cdot)r^T$$
$$\dot{\hat{V}} = -\kappa G \|r\| \hat{W} + Gx\left(\bar{\sigma}'^T(\cdot)\hat{W}r\right)^T$$

where $\kappa \in \mathbb{R}^+$ is selected to be a small constant, $F \in \mathbb{R}^{15\times 15}$, $G \in \mathbb{R}^{37\times 37}$ are positive definite gain matrices, $x(t)$ is the input vector defined in (27), and $r(t)$ is the filtered tracking error signal defined in (24). Here, the filtered tracking error signal $r(t)$ is required in the update laws which requires the measurement of the manipulator joint acceleration, and hence, this is undesirable. To ensure that the weights generated from this law are bounded, and that joint acceleration measurements are not required, we redefine the update laws as follows

$$\dot{\hat{W}} = -\alpha_w \hat{W} + \gamma_1 \bar{\sigma}\left(\hat{V}^T x\right) \ \text{sat}\left(e_2 + \zeta\right)^T \tag{31}$$

$$\dot{\hat{V}} = -\alpha_v \hat{V} + \gamma_2 x\left[\bar{\sigma}'\left(\hat{V}^T x\right)\hat{W}\text{sat}\left(e_2 + \zeta\right)\right]^T \tag{32}$$

where $\alpha_v$, $\alpha_w \in \mathbb{R}^+$ are small constants, $\gamma_1$, $\gamma_2 \in \mathbb{R}^+$ are control gains which effect the learning speed, the function $sat(\xi) : \mathbb{R}^{15} \mapsto \mathbb{R}^{15}$ was previously defined, and the auxiliary signal $\zeta(t) \in \mathbb{R}^9$ is a surrogate (i.e. a dirty derivative operation) for the signal $\dot{e}_2(t)$ which is defined as follows

$$\zeta = \frac{1}{\varepsilon}(e_2 - \eta) \tag{33}$$

[1]The number of neurons required for the system was determined experimentally by noting the performance achievable with a given number of neurons and increasing the number of neurons until satisfactory tracking performance was obtained.

where $\varepsilon \in \mathbb{R}^+$ is a small constant, and the signal $\eta(t) \in \mathbb{R}^9$ is updated according to the following expression

$$\dot{\eta} = \frac{1}{\varepsilon}(e_2 - \eta). \tag{34}$$

From equations (26)-(34) and the fact that the input vector to the neural network is bounded, it is easy to show that the weight matrices $\hat{W}(t)$ and $\hat{V}(t)$ are bounded, and hence, the output from the neural network, $\hat{f}(t)$, is bounded.

## V. Experimental Results

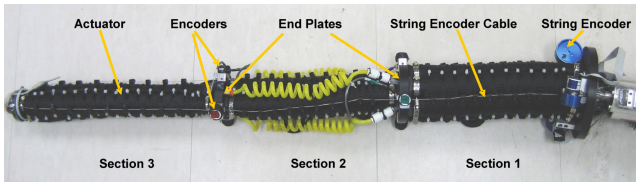



Fig. 1: The OCTARM V.2 robotic manipulator.

To verify the performance of the controller with the neural network feedforward component, the controller was implemented on the OCTARM continuum robot manipulator. In this section, we first provide a description of the OCTARM continuum robot manipulator, then experimental results are provided which demonstrate the effectiveness of the neural network feedforward tracking controller. Research work for the OCTARM robotic manipulators is being conducted for the Soft Robot Manipulators and Manipulations project supported by the DARPA BIODYNOTICS program. This work is a multidisciplinary and multi-institutional effort, the reader is referred to [2], [27], [28], for more details of the project. The team members at Penn State University perform the mechanical design and construction of the arms while the team at Clemson University develops the electronics, kinematics and control systems.

### A. Description of the OCTARM Manipulator

The OACTARM manipulator [28], [29] is a biologically inspired soft robot manipulator resembling a trunk or tentacle. The OCTARM is significantly more versatile and adaptable than conventional robotic manipulators, capable of adaptive and dynamic manipulation in unstructured environments. To provide the desired dexterity the OCTARM is constructed with high strain extensor air muscles called McKibben actuators, which are constructed by covering latex tubing with a double helical weave, plastic mesh sheath [29]. These actuators provide the large strength to weight ratio and strain required for soft robot manipulators.

The OCTARM is divided into three sections with each section capable of two axis bending and extension allowing nine total degrees of freedom. The arm is pneumatically actuated with a maximum pressure of 130 Psi through nine pressure control valves. To provide two-axis bending and extension, three control channels are selected for each section. Six actuators are used in each section one and two and three actuators are used in section three. The six actuator design has two actuators for each control channel and results in actuators located at a larger radius, corresponding to higher stiffness and load capacity. Three closely-spaced actuators provide high curvature for the distal section. To provide torsional motion, the OCTARM has been fitted with a D.C. motor at the base. In OCTARM V.2 (see Figure 1), the D.C. motor is directly coupled to the end plate of the base section through a reduction gear mechanism. This arrangement of the base motor limits the maximum rotation to 180 degrees.

For closed loop control of the OCTARM manipulator accurate shape sensing is essential. The shape of the manipulator can be inferred by measuring the length of each of the actuators on the OCTARM. To measure the length of each actuator, three string encoders are used. Figure 1 shows the three section OCTARM V.2 with the string encoders attached to the base of section 1 and optical encoders located at the end plates of section 1 and 2. The cables from each of the string encoders run the entire length of the arm through the optical encoders at the lower sections, as seen in Figure 1. This configuration enables the length of each of the actuators on the OCTARM manipulator to be determined. To obtain actuator velocity measurements, a variable structure velocity observer is utilized (see [15]).

The design of these manipulators is constantly being refined to provide stronger actuators, additional sensory information, newer capabilities and eliminate some of the problems with the previous designs. With the OCTARM V.2, due to the arrangement of the string encoders at the base section and the optical encoders at the end plates of the distal sections, there were a number of protrusions on the surface of the arm limiting its grasping capabilities. Also the air tubes for the distal sections were coiled on the outer surface of the arm, again limiting its grasping capabilities. Another problem faced with OCTARM V.2 was slippage of the string encoder cable at the distal sections which caused a loss of calibration. To address these issues, a new prototype of the arm has been developed called OCTARM VI (see Figure 2). OCTARM VI has also been fitted with a rotary union which has an electrical slip ring with 36 electrical connections and also provides 12 independent passages for pneumatic lines. This new rotary union enables continuous rotation of the base of the manipulator. Shape sensing with the string encoders has also been reconfigured in OCTARM VI. There are now nine string encoders arranged around the base section. New eyelets for guiding the encoder cables have also been developed to reduce friction. In addition, the electrical wiring for sensors and the air tubes for pneumatic channels have been enclosed inside the actuators, providing a clean exterior surface of the arm for grasping.

The robot control system consists of commercial off-the-shelf Pentium III EBX form-factor Single Board Computer (SBC) with two ServoToGo data acquisition boards which provide analog and digital I/O. The computer runs the QNX Neutrino real-time Operating System and QMotor [30] the in-house developed hard real-time control software for implementation of the control algorithms. Data acquisition and control implementation were performed at a frequency

of 500 Hz.

### B. Joint Trajectory Tracking Experiment Description

Preliminary experimental results were obtained using the OCTARM V.2 to demonstrate the effectiveness of the neural network feedforward control. To test the low level controller with the neural network component given in (25), a sinusoidal desired trajectory was selected for the actuator lengths. The three actuators on a section are 120 degrees apart mechanically, so the desired trajectory for each actuator in a section is shifted 120 degrees in phase from the trajectory of the previous actuator in that section. The trajectory for section $i$, where $i = 1, 2, 3$ represents the three sections of the OCTARM, was selected as follows

$$q_{dik} = l_{min_i} + (1 - \exp(-0.5t)) \left[ l_0 + 2\sin\left(0.0625\pi t + \frac{2}{3}\pi k\right)\right] \ \forall k = 1, 2, 3$$

where $q_{di} = [q_{di1}, q_{di2}, q_{di3}] \in \mathbb{R}^3$ represents the desired trajectory for the actuators on section $i$, $l_{min_i} \in \mathbb{R}$ represents the minimum lengths of the actuators on section $i$ and $l_0 = 7$ [cm] is the initial extension of the actuators on section $i$. The initial extension was selected to keep the operating pressure close to the center of its operational range. The minimum and maximum lengths of the sections which correspond to the minimum pressure (0 psi) and maximum pressure (130 psi) respectively are physical limitations of the actuators and were found to be $l_{min_1} = 22.9$ [cm], $l_{min_2} = 22.4$ [cm], $l_{min_3} = 27.9$ [cm], $l_{max_1} = 35.8$ [cm], $l_{max_2} = 37.1$ [cm], $l_{max_3} = 47.7$ [cm].

The system gains which yielded satisfactory performance were determined by trial and error and were as follows

$$K_s = diag\{1, 1, 1, 1, 1, 1, 1, 1, 1\},$$
$$\beta = diag\{1, 1, 1, 1, 1, 1, 0.5, 0.5, 0.5\},$$
$$\lambda_1 = 1, \ \ \lambda_2 = 1, \ \ \gamma_1 = 10, \ \ \gamma_2 = 500,$$
$$\alpha_v = 0.001, \ \ \alpha_w = 0.001, \ \ \varepsilon = 0.01.$$

There was no training period utilized to determine the initial values for the weight matrices $\hat{W}(t)$ and $\hat{V}(t)$, the matrices were initialized to zero.

### C. Analysis of Results

To test the effectiveness of the neural network feedforward term, we compared the controller in (25) with a standard PID controller and the controller given in (25) without the neural network component. To provide a means to quantify the performance of each controller, we compute the following measures

$$M_e \triangleq \int_{t_0}^{t} \|e_1(\tau)\|^2 d\tau \tag{35}$$

$$M_u \triangleq \int_{t_0}^{t} \|u(\tau)\|^2 d\tau \tag{36}$$

where $M_u(t)$ is a measure of the energy expended by the controller, and $M_e(t)$ is a measure of the magnitude of the tracking error over the period of operation of the system.

TABLE I: Comparison of Energy Measures for Different Controllers

| | **PID** | $u(t)$ **without** $\hat{f}(t)$ | $u(t)$ **with** $\hat{f}(t)$ |
|---|---|---|---|
| $M_e$ | $2.8357 \times 10^3$ | 256.2135 | 58.2223 |
| $M_u$ | $1.3689 \times 10^6$ | $2.0459 \times 10^6$ | $2.069 \times 10^6$ |

The performance of the system was first tested without the neural network component. The control gains for the controller were adjusted till good performance was obtained. Figures (3, 4, 5) show the actual and desired joint trajectories, joint tracking error, and the input pressure for the controller without the neural network component. Next, the neural network feedforward was added to the controller, and the neural network weight update law gains were adjusted till best performance was obtained. Figures (6, 7, 8) show the actual and desired joint trajectories, joint tracking error, and the input pressure for the controller with the neural network feedforward component. It can be seen from Figure 7 that the tracking error with the neural network feedforward component settles out to $\pm 0.5$ [cm].

To compare the controller performance with and without the neural network feedforward component, the energy measures were computed for the two configurations. The energy measures were also computed for a standard PID controller, these results are presented to show the performance improvement obtained by using the controller in (25). Table I, shows a comparison of the performance for the three controller configurations. It can clearly be seen from Table I that improved tracking performance is achieved by adding the neural network feedforward to the controller.

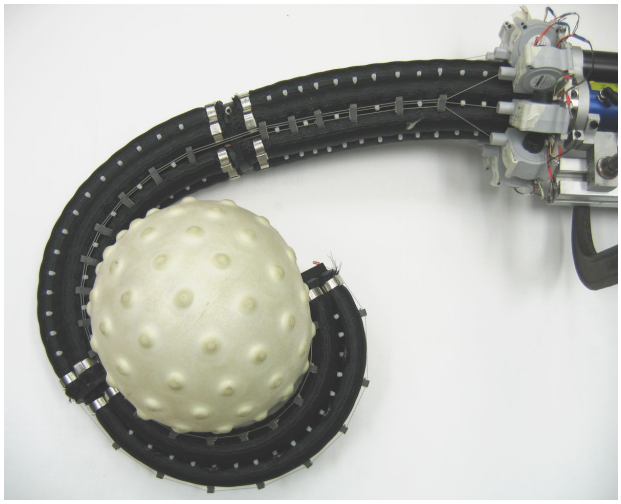


Fig. 2: OCTARM VI grasping a ball.

### D. Grasping Experiment Description

The whole arm grasping experiment with the neural network based joint controller will be conducted with OCTARM VI in the upcoming months. Figure 2, shows the OCTARM VI grasping a circular object. The object specific functions

for this planar application are defined as follows

$$\beta_i\left(X_i\right) \triangleq \left(x_i - x_c\right)^2 + \left(y_i - y_c\right)^2 - r_o^2 \; \forall \, i = 1, .., 6 \quad (37)$$

where $X_c = [x_c, y_c]^T \in \mathbb{R}^2$ represents the co-ordinates of the center of the object, and $r_0 \in \mathbb{R}$ represents the object radius. The task space variable for each of the three sections and the mid-point of the three sections are $X_i = [x_i, y_i]^T \in \mathbb{R}^2 \; \forall \, i = 1, .., 6$. The following task-space velocity field for a planar, circular contour will be utilized [26]

$$\dot{X}_d = \vartheta(X_6) = -2K(X_6)f(X_6) \begin{bmatrix} (x_6 - x_c) \\ (y_6 - y_c) \end{bmatrix} + 2c(X_6) \begin{bmatrix} -(y_6 - y_c) \\ (x_6 - x_c) \end{bmatrix} \quad (38)$$

where $\dot{X}_d \in \mathbb{R}^2$ represents the desired task space velocity for the end-effector, $X_6 \in \mathbb{R}^2$ represents the end-effector co-ordinate, and the functions $f(X_6)$, $K(X_6)$, and $c(X_6) \in \mathbb{R}$ are defined as in [26].

## VI. Conclusion

We have presented a neural network controller for grasping control of a continuum robot manipulator. The feedforward neural network was used to compensate for the uncertain nonlinear dynamics of the continuum manipulator, while the nonlinear controller provides semi-global asymptotic convergence of the joint position tracking error. Experimental results for the OCTARM soft robotic manipulator operating along a sinusoidal trajectory were presented. A comparison of the tracking performance, both with and without the neural network feedforward component demonstrates the efficacy of the proposed neural network feedforward estimation technique. Future work will consist of testing the whole arm grasping controller with the neural network component on OCTARM VI.

## References


[1] G. Robinson and J. B. C. Davies, "Continuum robots - a state of the art," in *Proc. IEEE Int. Conf. Robot. Automat.*, Detroit, Michigan, USA, 1999, pp. 2849–2854.

[2] I. D. Walker, D. M. Dawson, T. Flash, F. Grasso, R. Hanlon, B. Hochner, W. Kier, C. Pagano, C. D. Rahn, and Q. Zhang, "Continuum robot arms inspired by cephalopods," in *Proc. 2005 SPIE Conf. Unmanned Ground Vehicle Technology IV*, Orlando, Florida, USA, Mar. 2005, pp. 303–314.

[3] K. Salisbury, "Whole arm manipulation," in *Proc. 4th Int. Symposium Robotics Research*, 1987, pp. 183–189.

[4] K. Mirza and D. E. Orin, "Force distribution for power grasp in the digits system," *CSIM-IFToMM Symp, Theory and Practice of Robots and Manipulators*, 1990.

[5] P. Song, M. Yashima, and V. Kumar, "Dynamics and control of whole arm grasps," in *Proc. IEEE Int. Conf. Robot. Automat.*, Seoul, Korea, 2001, pp. 2229–2234.

[6] F. Asano, Z. Luo, M. Yamakita, and S. Hosoe, "Dynamic modeling and control for whole body manipulation," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, 2003, pp. 3162–3167.

[7] R. Platt, A. H. Fagg, and R. A. Grupen, "Extending fingertip grasping to whole body grasping," in *Proc. IEEE Int. Conf. Robot. Automat.*, Taipei, Taiwan, 2003, pp. 2677–2682.

[8] F. Matsuno and K. Suenga, "Control of redundant snake robot based on kinematic model," in *Proc. 41st SICE Annual Conference*, Osaka, Japan, 2002, pp. 1481–1486.

[9] M. Ivanescu, N. Popescu, and D. Popescu, "A variable length tentacle manipulator control system," in *Proc. IEEE Int. Conf. Robot. Automat.*, Barcelona, Spain, 2005, pp. 3285–3290.

[10] T. Suzuki, K. Shintani, and H. Mochiyama, "Control methods of hyper-flexible manipulators using their dynamical features," in *Proc. 41st SICE Annual Conference*, Osaka, Japan, 2002, pp. 1511–1516.

[11] F. Matsuno and H. Sato, "Trajectory tracking control of snake robots based on dynamic model," in *Proc. IEEE Int. Conf. Robot. Automat.*, Barcelona, Spain, 2005, pp. 3040–3045.

[12] M. Ivanescu, "Position dynamic control for a tentacle manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, Washington, DC, 2002, pp. 1531–1538.

[13] H. Mochiyama, E. Shimemura, and H. Kobayashi, "Control of manipulators with hyper degrees of freedom: shape tracking using only joint angle information," *International Journal of Systems Science*, vol. 30, no. 1, pp. 77–85, 1999.

[14] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. London: Taylor and Francis, June 1999.

[15] B. Xian, D. M. Dawson, and M. S. de. Queiroz, *A Continuous Asymptotic Tracking Control Strategy for Uncertain Nonlinear Systems: In Optimal Control, Stabilization, and Nonsmooth Analysis*, ser. Lecture Notes in Control and Information Sciences. Heidelberg, Germany: Springer-Verlag, 2004, vol. 301, pp. 251–264.

[16] C. Kwan, F. L. Lewis, and D. M. Dawson, "Robust neural-network control of rigid-link electrically driven robots," *IEEE Trans. Neural Networks*, vol. 9, no. 4, pp. 581–588, July 1998.

[17] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Networks*, vol. 7, no. 2, pp. 388–399, Mar. 1996.

[18] H. D. Patino, R. Carelli, and B. R. Kuchen, "Neural networks for advanced control of robot manipulators," *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 343–354, Mar. 2002.

[19] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," vol. 22, no. 1, pp. 43–55, Feb. 2006.

[20] I. Gravagne, C. Rahn, and I. Walker, "Large deflection dynamics and control for planar continuum robots," *IEEE/ASME Trans. Mechatron.*, vol. 8, no. 2, pp. 299–307, June 2003.

[21] H. Mochiyama and T. Suzuki, "Kinematics and dynamics of a cable-like hyper-flexible manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, Taipei, Taiwan, 2003, pp. 3672–3677.

[22] ——, "Dynamic modeling of a hyper-flexible manipulator," in *Proc. 41st SICE Annual Conference*, Osaka, Japan, 2002, pp. 1505–1510.

[23] D. Braganza, M. McIntyre, D. M. Dawson, and I. Walker, "Whole arm grasping control for redundant robot manipulators," Clemson University, CRB Technical Report CU/CRB/10/12/05/#1, http://www.ces.clemson.edu/ece/crb/publictn/tr.htm, Oct. 2005.

[24] Y. Nakamura, *Advanced Robotics Redundancy and Optimization*. Reading, MA: Addison-Wesley, 1991.

[25] P. Li and R. Horowitz, "Passive velocity field control of mechanical manipulators," *IEEE Trans. Robot. Automat.*, vol. 15, no. 4, pp. 751–763, 1999.

[26] I. Cervantes, R. Kelly, J. Alvarez-Ramirez, and J. Moreno, "A robust velocity field control," *IEEE Trans. Contr. Syst. Technol.*, vol. 10, no. 6, pp. 888–894, 2002.

[27] W. McMahan, B. Jones, I. Walker, V. Chitrakaran, A. Seshadri, and D. Dawson, "Robotic manipulators inspired by cephalopod limbs," in *CDEN Symposium on Biomimicry, Bionics and Biomechanics*, Montreal, Canada, 2004, pp. 1–10.

[28] W. McMahan, V. Chitrakaran, M. Csencsits, D. M. Dawson, I. D. Walker, B. Jones, M. Pritts, D. Dienno, M. Grissom, and C. Rahn, "Field trials and testing of the octarm continuum manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, Orlando, FL, 2006, to appear.

[29] M. B. Pritts and C. D. Rahn, "Design of an artificial muscle continuum robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, New Orleans, LA, USA, 2004, pp. 4742–4746.

[30] M. Loffler, N. Costescu, and D. M. Dawson, "Qmotor 3.0 and the qmotor robotic toolkit - an advanced pc-based real-time control platform," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 12–26, June 2002.

## APPENDIX
## EXPERIMENTAL FIGURES

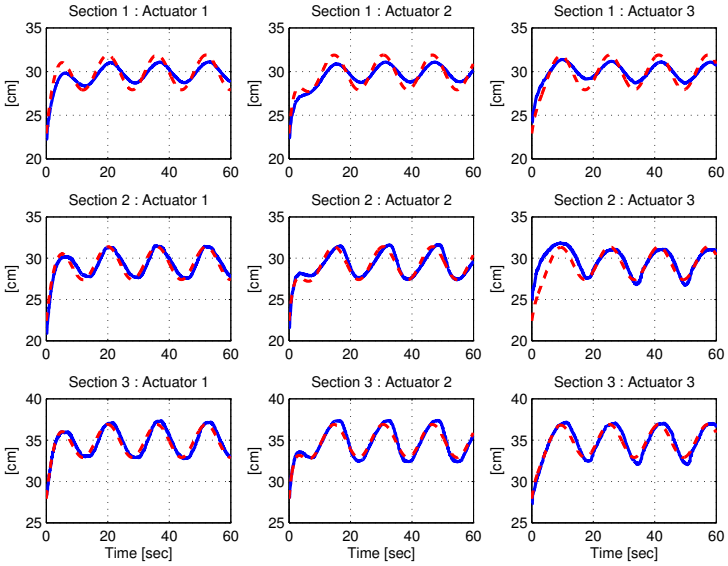



Fig. 3: Actual and desired joint trajectory without neural network component, solid line represents the actual joint trajectory, dashed line represents the desired joint trajectory.

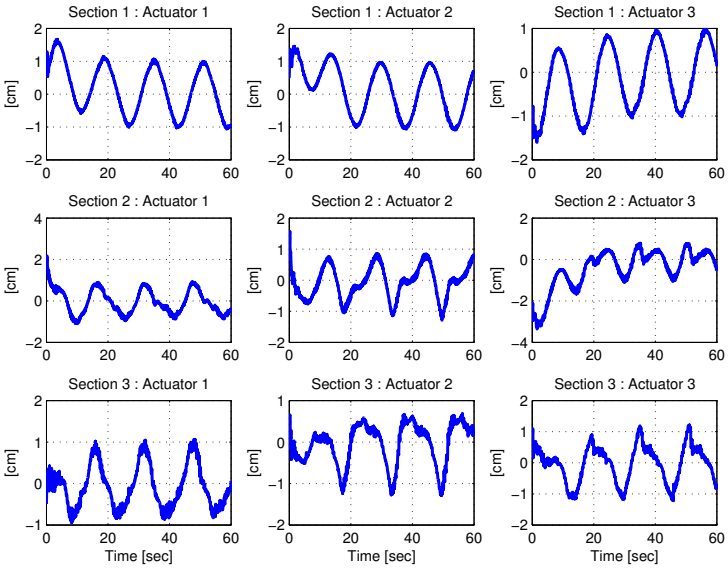



Fig. 4: Tracking error without neural network component.

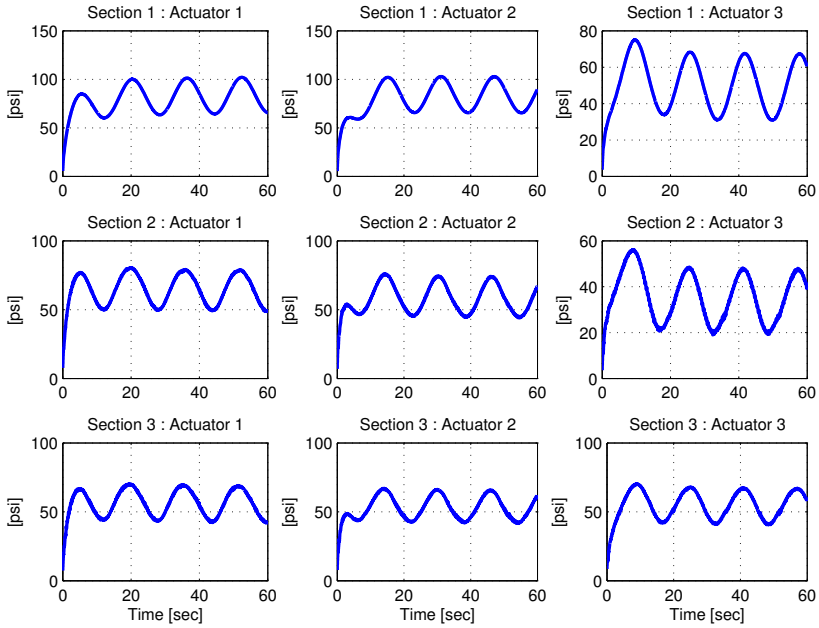



Fig. 5: Control pressure without neural network component.

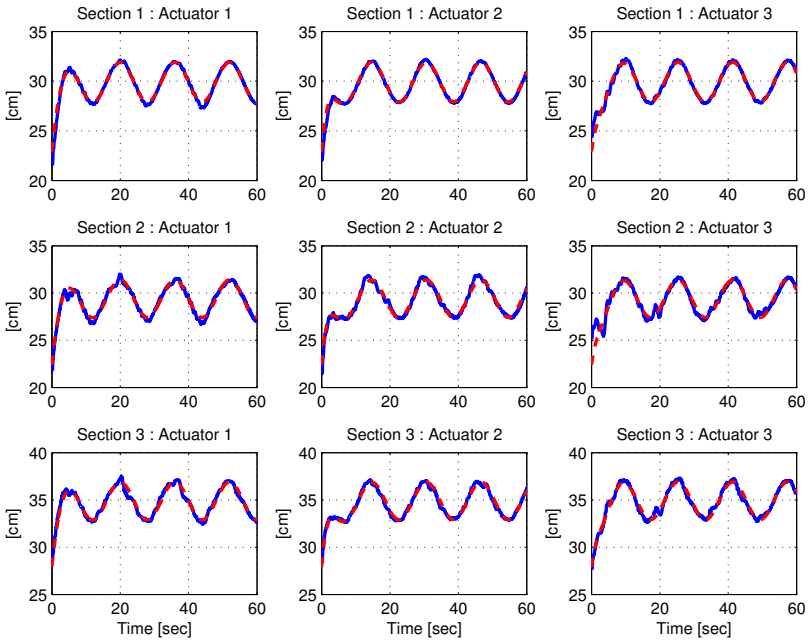



Fig. 6: Actual and desired joint trajectory with neural network component, solid line represents the actual joint trajectory, dashed line represents the desired joint trajectory.

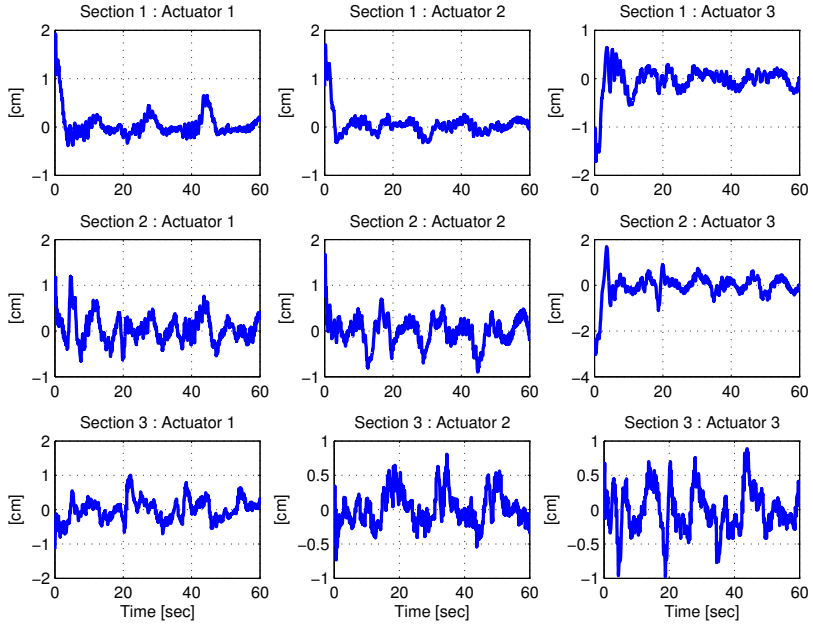



Fig. 7: Tracking error with neural network component.

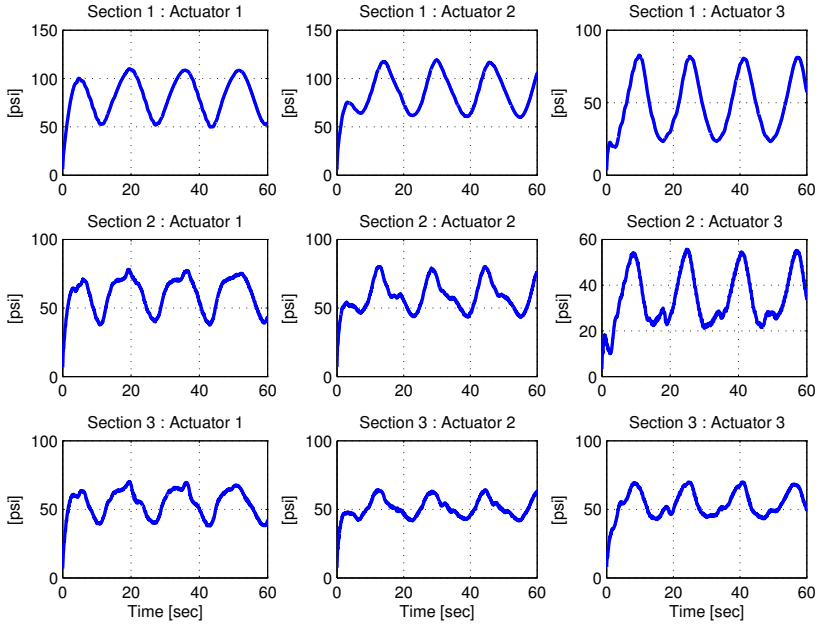



Fig. 8: Control pressure with neural network component.